Pattern Recognition Laboratory

PRTools Constructs

37 Steps

Datasets

A dataset combines the vector (feature) representation of objects with labels (class names), prior probabilities and possible other data annotation. `a`, `b` *and* `d` *are datasets. Almost all PRTools commands expect data to be supplied as a PRTools dataset.*

`a = dataset(data,labs)`	*define dataset from raw data*
`data = +a`	*convert dataset to double*
`s = struct(a)`	*inspect dataset fields or convert dataset to a structure*
`a = setlabels(a,labs)`	*(re)define labels*
`labels = getlabels(a)`	*retrieve labels*
`names = classnames(a)`	*retrieve class names*
`sizes = classsizes(a)`	*retireve classsizes*
`nlab = getnlab(a)`	*find object indices in class names*
`a = setprior(a,priors)`	*(re)set class priors*
`priors = getprior(a)`	*retrieve class priors*
`[m,k,c] = getsize(a)`	*get #objects, #features, #classes*
`d = [a b]`	*concatenate feature spaces*
`d = [a;b]`	*concatenate datasets*
`datasets`	*more info on datasets*
`multi_labeling`	*more info on multi-labeling system*

Datafiles

A datafile is a pre-stage of a dataset. It refers to objects organized as files (e.g. images) in directories and stores all pre-processing and feature definition needed to convert it to a dataset. Datafiles may bring large amounts of raw data (not yet normalized, varying sizes, no features extracted) within the domain of PRTools. Many commands defined for datasets apply to datafiles as well.

`a = datafile(dir)`	*define datafile from directory (dataset commands for labels and priors hold for datafiles as well)*
`struct(a)`	*convert datafile fields to struct*
`struct(a.dataset)`	*convert dataset fields in datafile*
`b = createdatafile(a,directory)`	*create new datafile from existing*
`a = filtm(a,command,par)`	*define preprocessing*
`b = dataset(a)`	*convert datafile to dataset*
`datafiles`	*more info on datafiles*

Sampling datasets or datafiles

`b = a(objects,:)`	*get subset of objects*
`b = a(:,features)`	*get subset of features (no datafiles)*
`b = selclass(a,name)`	*select one or more classes*
`[a,b] = gendat(a,siz)`	*random generation of subsets*

Mapping definition (in scripts and functions)

A mapping stores the definition of a mapping of one object representation (e.g. a vector space) into another. Some mappings may be trainable: they can be optimized for a given dataset.

`w = mapping(file,type,data,...)`	*low level routine to define mapping*
`args = setdefaults(argin,def1,...)`	*set defaults in mapping routine*
`w = define_mappings(args,type)`	*high level routine to define mapping in combination with* **`setdefaults`**
`data = getdata(w,field)`	*retrieve data field*
`mapping_task(argin,task)`	*test on mapping task*
`labels = getlabels(w)`	*retrieve labels*
`mappings`	*more info on mappings*

Mapping handling

Let `u` *be a trainable, yet untrained mapping and let* `v` *and* `w` *be trained or fixed (fully user specified) mappings.* `a`, `b` *and* `d` *are datasets.*

`w = a*u`	train `u` by `a`
`b = a*w`	*map* `a` *by* `w`
`v = w1*w2`	*sequential combination of mappings* `w1` *and* `w2`*:* `a*v = (a*w1)*w2`
`u = u1*u2`	*sequential combination of trainable mappings:* `a*u = a*u1*(a*(a*u1)*u2)`
`d = a*[v w]`	*same as* `d = [a*v a*w].v` *and* `w` *should be different mappings between the same representations*
`d = [a b]*[v;w]`	*same as* `d = [a*v a*w].v` *and* `w` *should be mappings to the same representation.*

Classifier handling

Classifiers are a special type of trainable mappings that map an object on class confidences or labels. Operations defined for mappings apply to classifiers as well. Let `u` *be an untrained classifier and let* `v` *and* `w` *be trained classifiers.* `a`, `b` *and* `d` *are datasets.*

`w = a*u`	*train* `u` *by* `a`
`b = a*w`	*map* `a` *by* `w` *on the classifier output space (e.g. densities)*
`v = w*classc`	*convert* `w` *into a classifier* `v` *that outputs confidences* `[0,1]`
`w = v*invsigm`	*convert a classifier* `v` *that outputs confidences into a classifier* `w` *that outputs distances* `[-inf,inf]`
`d = a*w*classc`	*same as* `d = a*(w*classc)`
`labout = a*w*labeld`	*classify dataset* `a` *and find labels*
`w = [w1 w2 ...]*maxc`	*stacked combining of classifiers, combined by* **`maxc`***, such that* `a*w = [a*w1 a*w2 ...]*maxc`
`d = a*[w1 w2...]*maxc`	*in* `d` *the outcomes of the individual classifiers are combined by the* **`maxc`** *rule. Many more exist.*
`w = [w1;w2;...]*maxc`	*parallel combining of classifiers, combined by* **`maxc`***, such that* `[a1 a2 ...]*w = [a1*w1;a2*w2;...]*maxc`
`d = [a1 a2 ...]*[w1;w2;...]*maxc`	*in* `d` *the outcomes of the individual classifiers are combined by the* **`maxc`** *rule.*

Globals

A set of global variables controls the behavior of PRTools.

`prglobal`	*list or reset globals*
`prmemory`	*defines the maximum size of internal variables; influences the number of loops.*
`prwaitbar`	*behavior of the waitbar*
`gridsize`	*resolution of plotting commands*
`defaultbatchsize`	*used by* **`setbatch`** *to control batch processing of large datasets.*

PRTools Procedures

Data generation

`circles3d`, `lines5d` *circles and lines*
`gendatb`, `gendatc`, `gendatd`, `gendath`, `gendatl`, `gendatm`, `gendats`, `spirals` *2D problems*
`gendatgauss`, `gentrunk` *multi-dim problems*
`gendat`, `gendatw`, `gensubsets` *generation of subsets*
`gendatk`, `gendatp` *interpolation*

Data import

`prdata` *load raw data, convert to dataset*
`prdataset` *load subset of dataset from matfile*
`prdatasets` *import public domain data*
`prdatafiles` *import public domain data as datafile*

Handling images

`data2im` *convert dataset to image*
`obj2feat`, `feat2obj` *object images <--> feature images*
`im2feat`, `im2obj` *image to feature or object in dataset*
`imsize` *retrieve size of specific image in datafile*
`im_patch` *find / generate patches in object images*
`band2obj` *convert image bands to objects in dataset*
`bandsel` *select image bands in dataset or datafile*
`selectim` *select image in multi-band object image*

Image operations

`classim` *classify image using a given classifier*
`doublem` *convert datafile images into double*
`filtim` *image operations for datafiles and datasets*
`spatm` *spatial smoothing of pixel classification*
`datunif`, `datgauss`, `im_box`, `im_fft`, `im_gray`, `im_label`, `im_maxf`, `im_minf`, `im_norm`, `im_resize`, `im_rotate`, `im_scale`, `im_select_blob`, `im_threshold`

Features from images

`histm`, `im_harris`, `im_moments`, `im_mean`, `im_measure`, `im_profile`, `im_stat`, `im_skel_meas`

Feature selection

`feateval` *evaluation of a feature set*
`featrank` *ranking of individual feature permormances*
`featsel` *user supplied feature selection*
`featseli`, `featselb`, `featself`, `featselo`, `featselp`, `featsellr`, `featselm` *various feature selection strategies*

Fixed mappings

`cmapm` *some special maps*
`sigm`, `invsigm` *(inverse) sigmoid map*
`filtm` *arbitrary operation on datafiles/datasets*
`normm` *object normalization*
`remoutl` *remove outliers*

Trainable mappings

`scalem` *find appropriate scaling*
`bhatm`, `fisherm`, `chernoffm`, `nlfisherm` *linear supervised mappings*
`klm`, `klms` *decorrelation and Karhunen Loève mapping*
`pca` *principal component analysis*
`proxm` *proximity mapping and kernel construction*
`reducm` *reduce to minimal space mapping*
`kernelm` *kernel mapping*
`userkernel` *user supplied kernel definition*
`gtm`, `som` *special mappings*

Density estimation

`gaussm` *mixture of Gaussians*
`knnm` *k-Nearest neighbor density*
`parzenm` *Parzen density*
`parzenml` *ml estimation of smoothing for Parzen*

Clustering and distances

`distm` *distance matrix between two datasets.*
`emclust` *expectation - maximization clustering*
`proxm` *proximity mapping and kernel construction*
`hclust` *hierarchical clustering*
`kcentres` *k-centres clustering*
`kmeans` *k-means clustering*
`modeseek` *clustering by modeseeking*
`mds`, `mds_cs` *multi-dimensional scaling*

Regression

`linearr`, `ridger`, `lassor`, `svmr`, `ksmoothr`, `knnr`, `pinvr`, `plsr`, `plsm`, `gpr`, `testr`, `rsquared`, `gendatr`

Classifiers, linear and quadratic

`fisherc`, `ldc`, `loglc`, `nmc`, `nmsc`, `qdc`, `udc`

Classifiers, support vector machine (svm)

`libsvc`, `nulibsvc`, `rblibsvc`, `pklibsvc` *based on the LIBSVM package*
`svc`, `nusvc`, `rbsvc` *PRTools based SVM*

Classifiers, neural net based

`bpxnc`, `lmnc`, `perlc`, `rbnc`, `rnnc`, `vpc`, `drbmc`

Classifiers, various

`mogc`, `parzenc`, `parzendc`, `nmsc`, `ldc`, `udc`, `qdc`, `naivebc` *density based classifiers*
`treec`, `dtc`, `randomforestc`, `stumpc` *decision trees*
`weakc`, `knnc`, `baggingc`, `adaboostc`, `fdsc` *other classifiers*

Combining classifiers

`averagec`, `dcsc`, `modselc`, `rsscc`, `votec`, `wvotec`, `maxc`, `minc`, `meanc`, `medianc`, `mlrc`, `naivebcc`, `perc`, `prodc`, `traincc`

Classifiers, related routines

`distmaha` *Mahalanobis distance more routines*
`meancov` *Estimation of means and covariances*
`edicon` *Edit and condense training sets*
`testk` *Error estimation for k-nearest neighbour rule*
`testp` *Error estimation for Parzen classifier*
`testn` *Error estimate for normal distributions*
`testc` *General error estimation routine*
`classc` *Converts a mapping into a classifier*
`labeld` *Find labels of objects by classification*
`rejectc` *Creates reject version of exisiting classifier*

Evaluation

`classim` *classify image using a given classifier*
`cleval` *classifier evaluation (learning curve)*
`clevalf` *classifier evaluation (feature size curve)*
`confmat` *computation of confusion matrix*
`costm` *cost mapping, classification using costs*
`crossval` *crossvalidation*
`disperror` *display annotated error matrix*
`labelim` *construct image of labeled pixels*
`loso` *leave_one_set_out crossvalidation*
`reject` *compute error-reject trade-off curve*
`roc` *receiver-operator curve (ROC)*
`shiftop` *shift operating point of classifier*
`testc` *general classifier error estimation routine*
`testd` *error of dataset applied to given classifier*
`testauc` *estimate error as area under the ROC*

Plot routines

`plotc`, `plotm` *plot classifier, mapping in scatterplot*
`plote` *plot error curves*
`plotf` *plot feature distribution*
`ploto` *plot object functions*
`plotdg` *plot dendrgram (see hclust)*
`scatterd`, `scatterdui` *scatterplots*
`show` *display objects (mainly for images)*

Examples

`prex_cleval`, `prex_combining`, `prex_confmat`, `prex_datafile`, `prex_datasets`, `prex_density`, `prex_eigenfaces`, `prex_matchlab`, `prex_mcplot`, `prex_plotc`, `prex_som`, `prex_spatm`, `prex_cost`, `prex_logdens`, `prex_soft`, `prex_regr`